2100030336

M. Gayathri Ankitha

.NET PROGRAMMING

LAB-7

IN-LAB

TASK1: To create generic type **CustomArray** – one dimensional array with random index range

CustomArray is a collection – array of random type values with fixed length and with original index that is specified by user.

Example 1: array of 20 elements length, array values – symbols, index starts with 18.
Example 2: array of 10 elements length, array values – objects of class Animals, index starts with -5

Values of random type can be located in array, custom first index and the number of elements in array should be specified while creating. The length and range of indexes cannot be changed after creating. Values of array elements can be set while creating the array and later with the help of indexer.

Initial and finite index, array length, array elements in the form of standard array Array that starts with 0 can be obtained from array.

CustomArray should be able to use operator foreach and other constructions that are oriented to the presence of numerator in class.

The task has two levels of complexity: Low and Advanced.

Low level tasks require implementation of the following functionality:

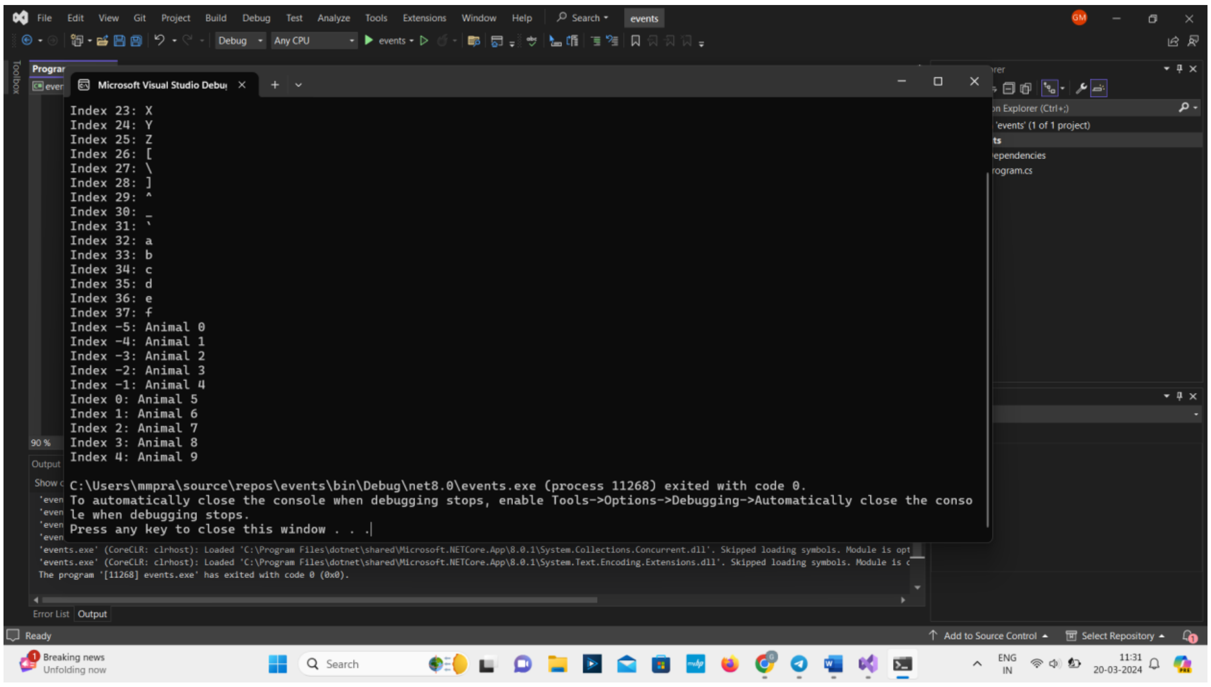
- Creating of empty user array and the one based on standard existing
- Receiving first, last indexes, length, values in form of standard array with 0.
- Access to writing and reading element based on predetermined correct index

Advanced level tasks require implementation of the following functionality:

- All completed tasks of Low level
- Creating of array based on values params
- Generating exceptions, specified in xml-comments to class methods
- Receiving numerator from array for operator foreach

```
using CustomArray;
using System;
using System.Collections;
using System.Collections.Generic;

namespace CustomArray
{
    /// <typeparam name="T">Type of elements in the array.</typeparam>
    public class CustomArray<T> : IEnumerable<T>
    {
        private T[] array;
        private int startIndex;
        /// <param name="length">Length of the array.</param>
        /// <param name="startIndex">Start index of the array.</param>
        public CustomArray(int length, int startIndex)
        {
            if (length <= 0)
                throw new ArgumentException("Length must be greater than zero.");

            this.array = new T[length];
            this.startIndex = startIndex;
        }
        /// <param name="values">Values to initialize the array with.</param>
        /// <param name="startIndex">Start index of the array.</param>
        public CustomArray(T[] values, int startIndex)
        {
            if (values == null)
                throw new ArgumentNullException(nameof(values));

            this.array = values;
            this.startIndex = startIndex;
        }
        public int Length => array.Length;
        public int FirstIndex => startIndex;
        public int LastIndex => startIndex + array.Length - 1;
        /// <param name="index">The index of the element to get or set.</param>
        /// <returns>The element at the specified index.</returns>
        public T this[int index]
        {
            get
            {
                CheckIndex(index);
                return array[index - startIndex];
            }
            set
            {
                CheckIndex(index);
                array[index - startIndex] = value;
            }
        }
        public T[] ToStandardArray()
        {
            T[] standardArray = new T[array.Length];
            Array.Copy(array, standardArray, array.Length);
            return standardArray;
        }
        public IEnumerator<T> GetEnumerator()
        {
            foreach (T item in array)
            {
                yield return item;
            }
        }
        IEnumerator IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
```

```csharp
        }
        /// <param name="index">Index to check.</param>
        private void CheckIndex(int index)
        {
            if (index < startIndex || index >= startIndex + array.Length)
                throw new IndexOutOfRangeException($"Index {index} is out of
range.");
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        CustomArray<char> charArray = new CustomArray<char>(20, 18);
        for (int i = charArray.FirstIndex; i <= charArray.LastIndex; i++)
        {
            charArray[i] = (char)(i + 65);
        }
        for (int i = charArray.FirstIndex; i <= charArray.LastIndex; i++)
        {
            Console.WriteLine($"Index {i}: {charArray[i]}");
        }
        Animal[] animals = new Animal[10];
        for (int i = 0; i < animals.Length; i++)
        {
            animals[i] = new Animal($"Animal {i}");
        }
        CustomArray<Animal> animalArray = new CustomArray<Animal>(animals, -5);
        for (int i = animalArray.FirstIndex; i <= animalArray.LastIndex; i++)
        {
            Console.WriteLine($"Index {i}: {animalArray[i].Name}");
        }
    }
}
class Animal
{
    public string Name { get; set; }
    public Animal(string name)
    {
        Name = name;
    }

}
```



```
Index 23: X
Index 24: Y
Index 25: Z
Index 26: [
Index 27: \
Index 28: ]
Index 29: ^
Index 30: _
Index 31: `
Index 32: a
Index 33: b
Index 34: c
Index 35: d
Index 36: e
Index 37: f
Index -5: Animal 0
Index -4: Animal 1
Index -3: Animal 2
Index -2: Animal 3
Index -1: Animal 4
Index 0: Animal 5
Index 1: Animal 6
Index 2: Animal 7
Index 3: Animal 8
Index 4: Animal 9

C:\Users\mmpra\source\repos\events\bin\Debug\net8.0\events.exe (process 11268) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
'events.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Collections.Concurrent.dll'. Skipped loading symbols. Module is opt
'events.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Text.Encoding.Extensions.dll'. Skipped loading symbols. Module is c
The program '[11268] events.exe' has exited with code 0 (0x0).
```